Data Modeling Guidelines

Overview

This paper is intended to provide guidelines in the development of logical data models. It is based on recognized industry standards and practices utilized in the development of logical (business) models and should be incorporated wherever possible. Additionally, these guidelines are not tool specific but are based on sound methodology practices. The purpose of these guidelines is to socialize a general approach to data modeling within SFA. This will ensure that similarly constructed models can be easily compared, categorized, grouped, stored and analyzed. This will help ensure that SFA maximizes development resources through reuse of shared metadata and data objects.

In this section you will find:

- Data Modeling/Entity Relationship Diagrams Defined
- Data Model Types and Participant Roles
- Forward/Reverse Engineering Data Models and Participant Roles
- Data Normalization Overview

I. Data Modeling/Entity Relationship Diagrams Defined

The goal of data modeling is to clearly convey the definition, interrelationships and characteristics of SFAs data. In the simplest sense it is a technique used to analyze and record the descriptive information (properties, characteristics, business rules, definition, etc.) about SFA data.

Data modeling involves discovering business data usage patterns within the organization. This information is then graphically represented in an Entity Relationship Diagram (ERD) with its descriptive information being documented in entities, attributes and relationships. The resultant ERD and its descriptive information should be stable over time and flexible to changing business requirements.

The ERD consists of entities, attributes and relationships. AN ERD should be developed:

- Through either facilitated group or one-on-one business interview sessions,
- By defining the logical data structure without regard to implementation,
- Considering that it must be presentable and understandable to both technical and non-technical individuals,
- As a graphical representation and constructed using a data modeling tool,
- Through detailed discussion, documentation and examples illustrating the use of data within the business environment.

II. Data Model Types and Participant Roles

At SFA, three distinct types of models are developed and maintained as follows:

Conceptual Enterprise Data Model (CEDM)
Logical Data Model
Physical Data Model

This section details these three types and provides and explanation of their intended use.

Conceptual Enterprise Data Model

The Conceptual enterprise Data Model is a consolidated logical model that is intended to reflect a common view of data across SFA. It is a dynamic and evolutionary knowledge base that incorporates the channel verified data analysis as documented in legacy system or IPT logical data models. The model represents information concepts in business terms independent of any technical considerations. It is intended to provide an enterprise view of data and to serve as a basis for subsequent logical data models.

Logical Data Model

Logical data modeling is a technique for clearly representing business information structures and rules. Data modeling defines a context for turning data into information. The data definitions and the description of the structure of the data resulting from the modeling, allows an organization to understand the function of each piece of data.

The data model is business user driven. The content and structure of the model are controlled by, the business client not the systems developer. The language used in the model is stated in business, not technical terms. The model will provide the users and developers with an excellent tool to understand the information used within the organization. It provides a clear specification of what is wanted and not necessarily what currently exists. It describes the information of the organizational needs to operate various parts of its business and specifies business rules that must be enforced by the information system. The model is not constrained by assumptions about what the underlying hardware and software can or cannot provide. The data structures resulting from the data modeling process are then used as input to the database design.

The Logical Data Model is developed during the Definition Stage of the SFA Solution Development Lifecycle (SDLC). It represents information concepts in business terms independent of any technical considerations. This model is initially populated from the CEDM. It inherits from the CEDM those data objects that are within the scope of the project or study undertaken.

The Logical Data Model is a fully normalized data model containing complete data analysis for the project or study. It represent the data concepts, verified by channel business users, for all information within the scope of the study independent of technical or automation considerations. It includes all of the subject areas, entities, attributes, relationships and properties to support defined business processes (this is sometimes referred to a fully-attributed data model). The Conceptual Enterprise Data Model is an example of a Logical Model. Upon completion this model is used to further enrich the CEDM and can provide the basis for an implemented Physical Data Model.

Anticipated Common Roles:
Role: Data Modeler

- Determines the inclusiveness of data to meet specific business processes down to the attribute transformation level.
- Places data in its most detailed and atomic level.
- Facilitates the investigation, identification, elimination or minimization of redundant data in models and application systems.
- Facilitates the investigation of business data integrity and data quality issues.
- Establishes stable but flexible data structures and relationships for business processing by placing data in third normal form.
- Facilitates the identification, maintenance and management of a linkage between CEDM defined and used business data (data sharing and reuse).
- Facilitates the development, maintenance and analysis of any and all mappings or linkages between the physical occurrences of data (specific data implementations or databases) and their logical representations.
- Identifies and documents required metadata such as business/technical stewards, authoritative sources, security levels, backup/archival/retention requirements for logical data.

Role: Systems Architects/Integrators/Analysts

- Reviews and plans for data integration across architectural environments, applications and application systems.
- Notes and plans for synergies between logical data structures and application systems.
- Provides the business process business rule validation for information in the logical data model.

Role: Subject Matter Experts

- Validates the specific entities, attributes and relationships that are present in order to meet specific business processing and business rule and integrity requirements.
- Validates the correctness and quality of data definitions and domains (eg.data values, formats, lengths)

Physical Data Model

The Physical Data Model represents the data concepts represented in a Logical Data Model that has been scoped within an automation boundary and has been constrained by the physical and performance characteristics of a specific hardware/software implementation. It may be denormalized due to physical implementation considerations. It also may include data objects that are required from a technical implementation standpoint. This includes data properties such as the target database, target programming language, processing speed, disk size and allocation. A data warehouse snowflake or star schema design would be an example of a Physical Data Model (once documented in a data model tool. The model may be the result of reverse engineering a database schema or Data Definition Language (DDL). It also may be forward engineered from a Logical Data Model to derive an initial Physical Data Model. Forward Engineering and Reverse engineering are described below.

The general purpose of the Physical Data Model is to:

- Illustrate the properties of a physical data model prior to creating or modifying a database schema.
- Provide a link between the business representation of entities, attributes and relationships found in a Logical Data Model and a database implementation.
- Provide a graphical representation of a physical database.
- Utilize a data modeling tool to generate or modify database SQL scripts.

III. Forward/Reverse Engineering Data Models and Participant Roles

Forward Engineering

The process of forward engineering moves a standard data object in a normalized logical data model into a Physical Data Model. The Physical Data model is then prepared for transformation to a DBMS schema. Forward engineering from a fully attributed data model provides consistency and correlation between the models.

Reverse Engineering

The purpose of reverse engineering is to create a Physical Data Model from the Data Definition Language (DDL) of an existing database. Usually a data base is re-engineered when a Logical Data Model is not available for a given database. Other reasons include:

- The system is a migration system that is not well structured or documented but is planned to be enhanced or modified to incorporate additional requirements.
- The system is a legacy system that is not well structured or documented and will be incorporated, replaced or interfaced to designated migration systems.
- The system process a significant amount of redundant data that causes data quality or performance problems and requires redesign.

The process of reverse engineering involves:

- Identification of the systems that are the targets of a reverse engineering efforts
- Physical reading of a DBMS instance by a reverse engineering data modeling tool resulting in a Physical Data Model managed by a data modeling tool.
- Data analysis of the Physical Data Model to determine the logical entities and attributes to populate a Logical Data Model

Anticipated Common Roles:

ROLE: Data Modelers

- Consult/review physical changes (such as denormalization) with DBA's to ensure that implementations adequately reflect business rules.
- Provide guidance and orientation to the DBA's in the business concepts contained in the resulting database.
- Share metadata with the DBA's such that physical performance/archival/backup activities can be undertaken.

ROLE: Systems Designers/Integrators

- Review and plan the integration of data with specific application systems

ROLE: Data Base Administrators

- Review and modify the physical data model(prior to DDL generation) to target a specific hardware/software environment.
- Review and modify the Physical Data Model to apply physical naming standards.
- Implement/modify a database using the data modeling tool DDL generation

ROLE: Subject Matter Experts

- Usually there is limited or no role for a subject matter expert at this juncture (assuming that business requirements have been adequately documented in a Logical Data Model).

IV. Data Normalization Overview

One of the primary techniques in data modeling is data normalization. The term *data normalization* refers to the way data elements are grouped together into record structures. It is the process of developing a structured, logical data design. The goal of formal normalization, loosely interpreted, is to ensure that there is only one way to know a fact. The normalization process removes from the model all structures that provide more than one way to know the same fact.The goals of normalization, from the business point of view, are to ensure that the correct business rules are recorded, that incorrect business assumptions are removed or revised and that the resulting data model can easily be modified. The technical process of normalization helps with this. In the end, it is the correctness of the business assertions that needs to be validated. This requires a collaborative effort by users, analysts and technical personnel.

The objective of effective data normalization is to achieve at least third normal form. Other objectives of normalization include:

- Minimize the storage of redundant information.
- Build a data structure that is independent of the hardware and software used.

V. Guidelines for Data Model Components

These guidelines have been organized around the following data model components of a Logical Data model:

Subject Areas
Entities
Attributes
Relationships

These objects may be further broken down into sub-topics as they pertain to the above listed objects being discussed.

Data Model Components

Subject Areas
Descriptions

The description should define and distinguish it from all other subject areas from a business perspective. It should not reference or define past, current, or future organizational structures. A subject area must have at least one entity within it.

Entities
Descriptions

An entity definition must describe the scope and qualification of the entity type. The definition should be expressed in terminology common to the business and should include an indication of exactly what the entities are and how they are distinguished from one another. It should describe one occurrence of the entity. Entity descriptions should never imply multiple concepts, if this occurs it should most likely be separated into one or more entity or subtype. An example of this might be an entity type called PERSON and having the description read: An individual who is either an EMPLOYEE <u>or</u> CLIENT.

Volumetric

The statistical details of the entity should be specified such as the estimated number of occurrences as well as the estimates of increase or decrease over time.

Types of:

Core

A core entity is a major resource, product, or activity of interest to the business. Core entities are usually of a tangible or conceptual nature. Tangible entities might be CUSTOMER, PRODUCT, or EMPLOYEE. Conceptual entities are less tangible and might include such concepts as MARKET, QUOTE, or TRADE. Core entities are usually the focus or core of a subject area.

Subtype
An entity subtype is a more restrictive view of an entity type where occurrences of the entity type may have differing characteristics. Subtyping provides the ability to record additional predicates beyond those in the common set. Subtypes are defined through the use of a classifying or partitioning attribute in the Entity type "Supertype" and sets up for control of mutually exclusive groups of entities within the Supertype. Subtyped entities are said to inherit the predicates of their supertypes.

Associative
Since only entities can or should have attributes, relationship memberships are considered predicates of the entity types and not as objects in their own right. Thus a relationship can never have an attribute. Since a relationship has exactly two memberships two and only two entities can participate in a single given pairing. Models that enforce this constraint are said to depict binary relationships. When a relationship appears to require more than two memberships it should be replaced by an entity. When an entity is created for this reason it is called an associative entity. Associative entities may have relational pairings with more than two entity are then typically referred to as "super associatives"

Historical
This type of entity provides the supportive details of change to one or more of the attributes that are the predicates of the higher-level entity to which it is paired. When change history must be maintained relative to an entity type these details should be "pulled out" of the entity type and maintained separately in one or more related entities. The sole purpose of these types of entities are to record and maintain those details of change as they occur relative to an occurrence of the "parent entity".

Reference
Reference entities or "Look Up Tables" are just that, they provide the details of reference as they relate to other entities. These entities should be defined whenever the value of an attribute may provide or require additional details. Reference type entities should never be isolated or unrelated to the entities that reference them

Designer Added
Designer added entities should never occur within a Logical Data Model. Designer added entities are intended to over come processing constraints or to allow for deviations from the norm. This type of entity should be postponed as an implementation consideration.

Isolated
Isolated entities are entities which have no relational details maintained about them or their role or relationship with the business entities in the model. For exactly that reason they should not occur as there is a loss of business information. They are typically created as reference sets that provide details relating to attributes being maintained in other entity types. Most analysts who create these types of situations do so due to implementation considerations and in some cases to reduce the numbers of objects they should be required to maintain. Since there is a true loss of business detail these should not occur.

Attributes

Descriptions

An attribute description should include any textual information the analyst deems useful and should include information about its role and optionality.

Properties:

Optionality -
Simply states rather the attribute must or may have a value in an occurrence of that type of entity.

Domain
The domain of an attribute may be one of four types; it may be a text field, number, date, or time. It simply defines the type and format of the attribute value being described.

Length
The length of an attribute describes the number of characters or numbers it will allow to be recorded as a value in its occurrence. An attribute may be described as having either a fixed length or variable length depending on the business requirement.

Permitted Values
Permitted values are just that, they are the values that the attribute is intended to support and will allow no other values to be recorded in its occurrence. These should always be specified when it is necessary to restrict what the attribute may record.

Default Value
Default values are specified as necessary and describe what will be recorded as an attributes value if no other value is entered.

Identifying Types:

Natural Business
Natural Business Identifiers are those predicates that are readily recognized by the business and provide a natural uniqueness in the entity types occurrence. By choosing the natural value, which the business recognizes as being unique in its identification of the occurrence of the entity, the values provided are considered to be immutable which promotes the very essence of uniqueness. Examples of natural business identifiers might be a 'PRODUCT CODE' that is used to identify an instance of the entity type of 'PRODUCT'. These types of identifiers are readily understood by the business where choosing a non-business identifier such as system generated identifier would mean nothing to the business even though it would still provide the uniqueness required to record the occurrence. In those instances that a natural identifier cannot be identified then a sequential number may be used to uniquely identify an occurrence of an entity. This practice should be reserved to an absolute minimum.

Partitioning
Partitioning Identifiers are those classifying attributes used to identify and provide uniqueness to an entity types subtypes. Since an attribute may be of one and only one value at any given time it is therefore unique to the entities occurrence of that type. It is the basis for subdividing the entities of one type into subtypes. Subtypes within a

partitioning are mutually exclusive with each other but not with subtypes in other partitionings since subtypes may have partitionings of its own.

Sequencing
Sequence numbering of an entity implies a repetitive occurrence of the same type of occurrence that has a relationship to other occurrences of that entity type but must follow a sequencing strategy. A good example of this is in a line-numbering scheme of an ORDER LINE and its relationship to the other ORDER LINE occurrences of an ORDER entity type.

Supportive:

Dates
Dates are included as supportive attributions in the sense of recording the details of when an occurrence might occur. Dates such as creation, effectiveness, and no longer in effect such as cancellation dates are used to identify the effectiveness of the entity occurrence in question.

Status
Status concepts are those used to represent an entity types occurrence in its present ‘state’ within a life-cycle concept. Since an entity may exist in several states (one at a time) it provides the means to identify the condition of the entity types occurrence information and it’s relevance to the business. An example of this might relate to the purchase of an item that might be: Ordered, Shipped, or Received. The change of value in this status attribute must be a direct result of some processing action.

Multi-Valued
Sometimes an attribute seems to require the ability to maintain multiple values simultaneously and would be referred to as a multi-valued attribute. These type of attributes should be removed from the entity to which it belongs and promoted to an entity type of its own and then paired back to the originating entity.

Derived
Derived attributes are attributes that “derive” their value through some derivation algorithm that calculates its value based on the values of other predicates. These attributes require an execution of that derivation algorithm anytime the value of one of its base predicates changes in order to provide the corrected valuation. Derived attributes should not be used as identifiers for exactly this reason.

Designer Added
These types of attributes are invented to overcome some sort of business constraint or to simplify a system operation. They should not be incorporated in a business conceptual model as they are intended to facilitate its implementation.

Indicators
An indicator is used to express a two state yes/no informational condition. If an attribute in fact has more than two states it should be stated either as a set of permitted values relating to a STATUS or should be promoted as an entity type itself.

Relationships

Descriptions

A description should always be provided which provides clarity to the relational pairing that is being defined. One description each is required at the subject (source) and object (destination) ends of the pairing. The description should provide any details that help to define any rules that should be considered when making an actual pairing between the two entity occurrences.

Cardinality

Cardinality defines the number of entities from side that may be paired with a number of entities on the other side. Cardinality simply defines rather one entity occurrence may be considered in the pairing or many.

Optionality

Optionality specifies rather the defined cardinality (number of) entities from one side of the pairing may (optional) or must (mandatory) participate. If the cardinality specified is a 'many' and the optionality is mandatory, then at least one entity occurrence must be paired.

Volumetric

A number of volumetrics are desirable when available. These volumetrics are used to help to determine the physical environment requirements necessary to support the recording and storage of the information we are defining. The volumetrics that we seek are:

Estimated Percentage of Involvement (optional)
Estimated Number of Pairings (many)

Relationship Types:

Involuted (recursive)

An involuted or recursive relationship occurs when an entity of one type is paired with one or more occurrences of the same type. This typically occurs in the depiction of a parent-to-child relationship. All involuted relationships should be examined to assure that there is no loss of business information in the pairing. Most often these types of relationships may be resolved by defining a true hierarchical structure. These types of relationships should also be examined for the possibility of associative entity further supporting the pairing between the occurrences of the entity type involved.

Redundant

A redundant relationship is one that provides no information that cannot be deduced from other relationships. These types of relationship should be removed as superfluous since they will require duplicate operations with no benefit.

Fully Mandatory

Fully mandatory relationships, those being described as mandatorily paired on both ends of the relationship pairing, should be further examined. The issue to test the validity of a fully mandatory relationship is to ask "Does the occurrence of the first entity arise at exactly the same time as the second entity?".

Many-to-Many
Many to many relationships can occur frequently in the development of a logical model. The general rule is to resolve this anomaly by inserting an associative entity between the two related entity types. Usually, upon examination of the relational pairing, there are additional details about the pairing that should be maintained as business detail. All many-to-many relationships should be eliminated and resolved via associative entities.

Mutually Exclusive
Mutually Exclusive relationships provide a further refinement of detail as to the relational pairings that <u>may</u> take place between the occurrences of one type of entity and the occurrences of other entities types. This simply becomes the OR operator when the occurrences are to be paired. The mutually exclusive condition states that when associating a occurrence of one entity type it may be associated to the occurrences of one and only one of the other possible entity types stipulated in the mutually exclusive set. The example would be that the occurrences of Entity 'A' may be associated to either the occurrences of Entity 'B' or 'C' but can not be associated to the occurrences of both at the same time. This says that Entity types 'B' and 'C' are mutually exclusive to Entity 'A'.

Mutually Contingent
Mutually Contingent relationships are the exact opposite of mutually exclusive relationships. They state that when a pairing that occurs between one set of entity occurrences they MUST ALSO be paired with another set of specified entity types occurrences. In the example of Entity 'A', 'B', and 'C' where Entity 'A' has an occurrence paired with an occurrence of Entity 'B' it must also pair with a occurrence of Entity 'C'. What this says is that the pairing between 'A' and 'B' is contingent on the pairing between 'A' and 'C' as well.

VI. Special Topics

Additional Documentation Details

There may be a desire to record certain details or documentation concepts in the descriptive areas of the data objects, and with few exceptions, this should be avoided. If it is already documented do not re-document it elsewhere. The use of keyword searches through the descriptive areas of data object types when looking for such concepts as attribute domains, permitted values, default values, etc. These are all redundant documentation details. A good description for any data object should begin with the "Label" <u>BEN:</u> (**B**usiness **E**nglish **N**ame) followed with the true business English name for the object type. This allows for name abbreviations without loosing the full meaning. The only other possible "keyword" types of any value would fall in the categories of NOTES, EXAMPLES, and ALIAS/SYNONYM. Nothing should ever have the potential for requiring an update to its valuation